VrmlPad v. 2.0

User's Guide

Copyright © 1999-2002 ParallelGraphics

Contents

OVERVIEW 1

About VrmlPad 1
Product Features 1
New Features in VrmlPad 2.0 2
New Features in VrmlPad 1.3 3
Available Add-ins 4
Material Editor 4
Find in Files 4
Fields validation macro 5
Sample macros 5
Extrusion Editor 5
About VRML 6
Design Criteria 6
Characteristics of VRML 7
Scope 7
Introduction to the VrmlPad Automation 8

TEXT EDITOR 10

Overview: Text Editor 10
Syntax Coloring 10
AutoComplete 11
Error Processing 12
Node Folding 13
Finding Text 14
Regular Expressions 15
File Navigation 16
Using Drag-and-Drop Editing 17
Editor Commands and Keystrokes 17

SCENE TREE 20

Overview: Scene Tree 20
Scene Tree Navigation 21
Moving, Copying, and Cloning Nodes within the Scene Tree 21
Naming Nodes in the Scene Tree 22

ROUTING MAP 24

Overview: Routing Map 24
Managing Routes 24

RESOURCE VIEW 26

Overview: Resource View 26
Managing Resources 26
Using Drag-and-Drop in the Resource View 27

FILE LIST 29

Overview: File List 29

Using Drag-and-Drop in the File List 29

SCRIPT DEBUGGER 31

Overview: Script Debugger 31
Edit Time, Run Time, and Break Mode 31
Debugger commands 33
Debugger windows 35
Spreadsheet Fields 36
Viewing the Value of a Variable 37
Modifying the Value of a Variable 38

PUBLISHING WIZARD 39

The Basics of Publishing 39
Setting Up the Destination 39
Choosing Additional Resource Directories 39
Excluding Files from Publishing 40
VRML Optimization Options 40
Specifying Directory Organization 41
Reviewing the Directory Structure 41
Previewing Published Documents 41

HINTS AND TIPS 42

How to compress a VRML file 42
How to download a VRML file from the Net. 42
How to upload a VRML file to a remote server. 43
How to quickly find the definition (reference) of an identifier. 43
How to insert a pair of node or PROTO braces. 44
How to insert default field value. 44
How to indent a block of lines. 44
How to comment a block of lines or an entire node. 44
How to locate a syntax or semantic error. 45
How to make a VRML file smaller. 45
How to expose a field to a PROTO interface. 45
How to register a VRML extension. 46
How to change a set of the standard nodes. 46
How to change a source of the Node Help. 47
What are the ways I can access the VrmlPad object model? 47
How to automate VrmlPad from another application. 48

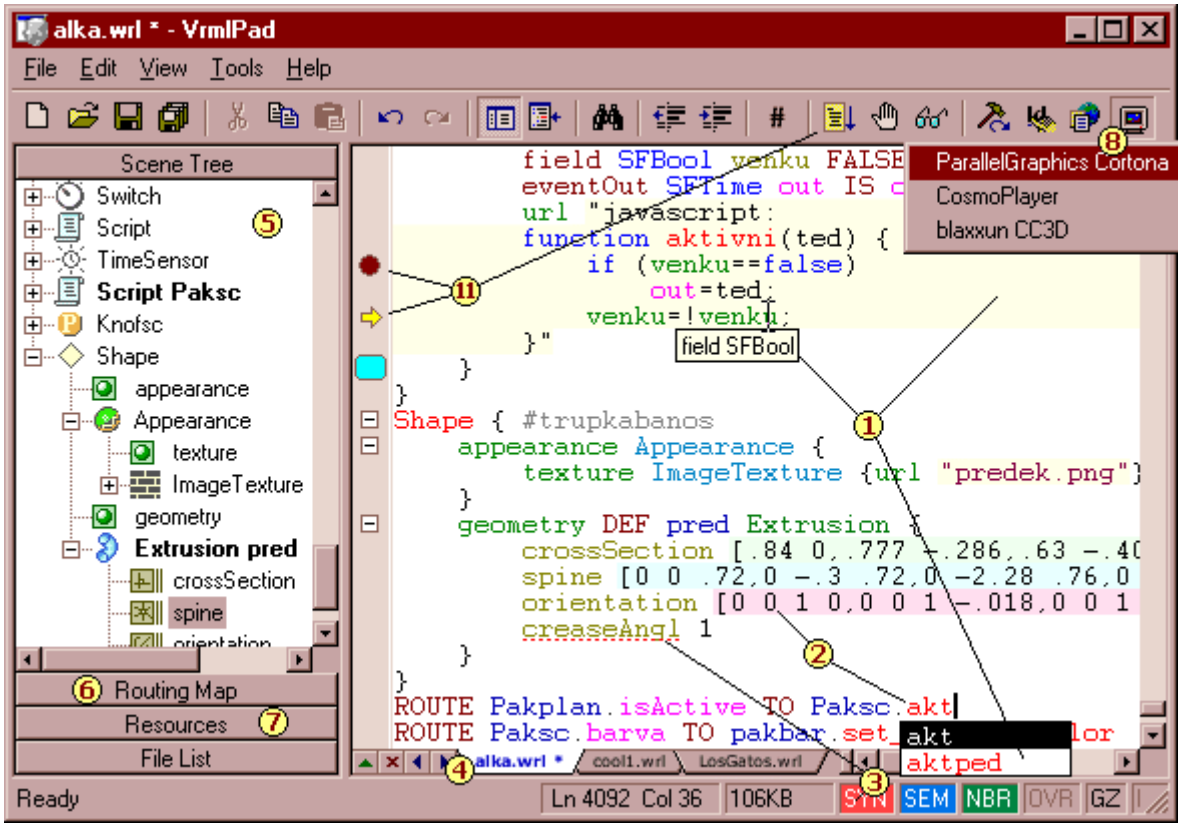
Overview

About VrmlPad

VrmlPad is a powerful and flexible authoring tool from ParallelGraphics that allows you to design and develop professional VRML content.

Use VrmlPad to create VRML worlds for publishing on the World Wide Web. VrmlPad fully supports the VRML 97 specification.

Product Features



① **Editing** – Access to local and remote files, multiple undo/redo, set bookmarks, advanced find and replace, syntax tips, smart AutoComplete, autoindent, dynamic node folding, customizable keyboard shortcuts.

② **Enhanced color-coding** – Customizable by VRML keywords, field and node categories.

③ **Dynamic errors detecting** – View and enumerate syntax and wide range of semantic errors and warnings – undefined identifiers, nonmatched fields, duplicated node definitions and more.

④ **Document Tabs** – Open and edit multiple documents. Find and replace in all opened documents. Document list includes also file dependencies of the active document.

⑤ **Scene Tree** – View, edit and navigate hierarchical scene structure. Synchronize selection in both directions. Delete, comment and rename nodes, PROTOs and field declarations.

⑥ **Routing Map** – View and edit scene routes. Synchronize selection in both directions. Delete, comment and add new routes using **Start Route** and **Add Route** commands.

⑦ **Resource View** – View, edit and navigate document dependencies. Rename and browse all references to a resource. Convert, when possible, absolute references to relative and vice versa.

⑧ **Preview** – Preview scenes in installed browsers and VRML viewers.

⑨ **Publishing Wizard** – (not shown on the figure) Organize and optimize your scene with dependencies for publishing on the Net and put the files on your Web server or send it by e-mail.

⑩ **Automation and Scripting** – (not shown on the figure) Manipulate the program from external applications using the C, C++, Visual Basic®, Borland Delphi®. Create add-ins, VBScript and JavaScript macros and execute them from within the VrmlPad environment, automating custom tasks.

⑪ **VrmlScript Debugger** – Locate bugs in your vrmlscript driven interactive VRML scenes using the integrated script debugger.

The debugger interface provides wrapper for the Cortona VRML client, special menus, windows, dialog boxes, and spreadsheet fields. Drag-and-drop functionality is available for moving debug information between components. Occasionally the debugger is paused in break mode, meaning the debugger is waiting for user input after completing a debugging command (like break at breakpoint, step into/over/out/to cursor, break after Break command or Restart)..

New Features in VrmlPad 2.0

Integrated Script Debugger

Now you can use debugging features that are collectively referred to as integrated script debugger. These features let you find and fix errors in the inline vrmlscript code in your VRML scenes and on HTML pages with embedded Cortona controls. The integrated script debugger is a full-featured debugger that enables you to:

- Control the execution of your script, including pausing the execution; running to a cursor, or a breakpoint location; stepping through code.
- Monitor the state of script data, including the current call stack; variables, arrays, and objects that are available in the current context; evaluate expressions, containing these data items.
- Modify the values of variables, arrays, and objects in your script during a debugging session.
- Add new procedures to your script during a debugging session as you would in code.
- Examine and modify on the fly data items in the script using the Quick watch window.
- Use CodeTips for quick viewing of variables values in the script.
- Examine full VRML text of runtime nodes corresponds to value of an expression.

Fields validation macro

The macro validates conformance of field values to the VRML97 specification. Some of about 100 checks performed by the macro: verifies node fields contain correct type of node or PROTO; increasing interpolators key values, ratio of key to keyValue entries; correct relationship between coord/coordIndex, texCoord/texCoordIndex, color/colorIndex, normal/normalIndex; Color values being in range 0 to 1 and more.

Search In Files add-in

The Find in Files window enables you to perform advanced text search in multiple VRML files a time (including compressed files). Literal text strings or regular expressions to find words or characters can be used in this command.

See also: "New Features in VrmlPad 1.3".

New Features in VrmlPad 1.3

Multiple documents

Now you can open multiple files in the same instance of VrmlPad, select between them by using the tabbed Document Bar and Ctrl+Tab hotkey, find and replace text in all opened files. Using the pull-down Document List you can browse opened files and VRML-dependencies (inlines, anchors and EXTERNPROTOs) of the active document.

Routing Map

The Routing Map displays routes of your scene and lets you browse and edit connections between events and exposed fields in your scene. Also you can more quickly add new routes to a scene using new Add Route dialog.

VRML extension node libraries

Now you can use the latest VRML extension nodes from Web3D consortium, ParallelGraphics, Blaxxun, GeoVRML, etc. VrmlPad supports semantic and publishing rules for the extension nodes and optionally insert appropriate EXTERNPROTO declarations for these nodes. The list of node libraries can be updated online. Also you can use your own custom set of extension nodes. To turn on some node extension, select appropriate checkmark from the Node libraries list at the General tab of the Options dialog box and restart VrmlPad.

New automation objects

Additional automation objects have been included: Documents and VrmlMatrix. The Documents collection represents all open workspace documents. The VrmlMatrix object provides many useful methods for performing manipulations on 4x4 matrices.

AutoComplete feature in inline scripts

In addition to list interface fields and events of a Script node, the Complete command (Ctrl+Space) now shows subfields of *SFNode* and *MFNode* fields, if possible, and lists names of standard VRML-script objects, properties and methods.

Line numbering

Additional margin at the left side of Text editor displays line number for every line of a document. To turn on line numbering, select Show line numbers checkmark at the Editor tab of the Options dialog box.

More Unused Identifiers

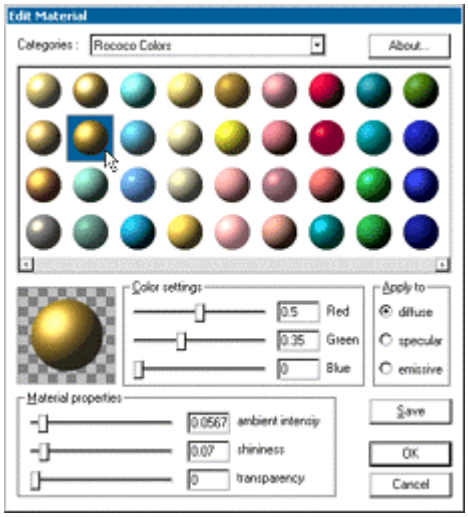
Now the Unused Identifiers dialog box shows also redundant (duplicated) field definitions and duplicated routes.

Improved performance

VrmlPad now parses, formats and indents/unindents text faster than previous version does.

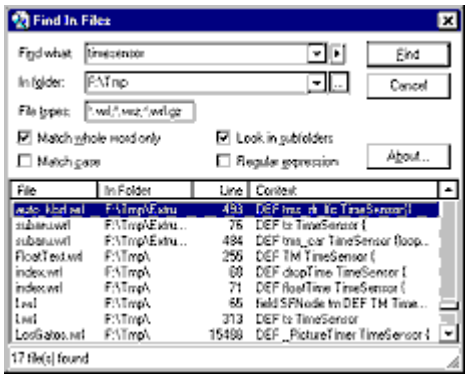
Available Add-ins

Material Editor



The Material Editor lets you create and edit Appearance and Material nodes. You also can choose existing materials from a pre-built library and add your own materials to it.

Find in Files



The Find in Files add- in enables you to perform advanced text search in multiple VRML files a time (including compressed files). Literal text strings or regular expressions to find words or characters can be used in this command.

To find a text string in multiple source files:

1. From the **Tools** menu, choose **Find in Files**.
2. In the **Find what** box, type the search text or a regular expression.
3. In the **In folder** box, select the folder that you want to search.
4. If necessary, select one or more of the **Find** options.
5. Click the **Find** button to begin the search.
6. To open a file containing a match, double-click the entry in the list below.

Fields validation macro

The Fields Validation macro checks scenes for conformance of various field values to the VRML97 specification. Among of about 100 tests performed by the macro are checks for: the field type compatibility (fields should contain nodes or PROTOs of correct type); increasing interpolators key values; ratio of key to keyValue entries; relationship between coord/coordIndex, texCoord/texCoordIndex, color/colorIndex, normal/normalIndex values; Color values being in range 0 to 1; etc.

To validate fields:

- From the **Tools** menu, choose **Validate Fields**.

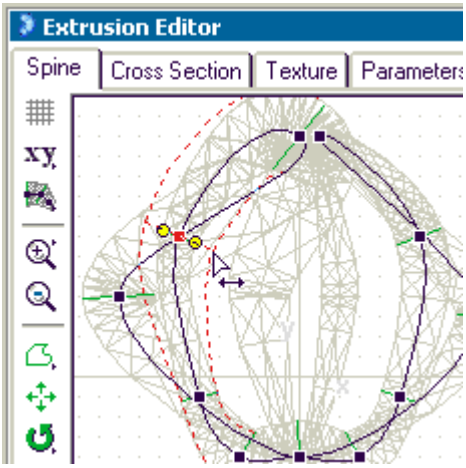
Sample macros

Sample macros include useful commands for enumerating faces in IndexedFaceSet nodes, completing selected nodes with all fields, and converting the Box, Cone and Cylinder nodes to an IndexedFaceSet etc.

Macro	Description
Complete All	Inserts all fields of the selected node with default values.
Go To Node	Prompts for a node name and selects the specified node.
Count Faces	Enumerates all faces in the scene and in the selected faceset.
WrapNodeBy…	Wraps the selected node by Group, Transform or Anchor nodes.
ConvertToFaceset	Converts selected Box, Cone or Cylinder node to IndexedFaceSet.
ScaleFacesets	Scales all facesets in the selected PROTO or the scene.

If you have not installed the macros during initial installation of VrmlPad, run the VrmlPad installation pack again, select the Sample macros checkbox and click OK.

Extrusion Editor



Extrusion Editor, a visual plug-in for VrmlPad, offers you an effective method for creating and editing existing extrusion models.

Extrusion Editor is not included in the VrmlPad installation pack. The trial version is available from the page http://www.parallelgraphics.com/products/vrmlpad/extrusioneditor/download/.

Key Features

- Drawing a series of connected vertices one by one
- Drawing Bezier curves (spline curves)

- The use of pre-defined forms (rectangle, diamond, octagon, triangle, oval, etc)
- Move, scale, flip and rotate the spine and cross-section
- Browse for, move, scale, and rotate the texture
- Assemble extrusions in complex models and perform editing operations on the group of shapes
- Manipulate extrusion parameters, edit the material, and control the smoothness of the surface
- Immediate preview changes in the 3D window
- Add new extrusions to the library of extrusions and insert their code into VrmlPad's text editor
- Full VRML97-compliance

Limitations in the trial version:

- 30-day evaluation period.
- You cannot save an extrusion (or a group of extrusions) if the number of segments in the spine is over 15.

See also: http://www.parallelgraphics.com/products/vrmlpad/extrusioneditor/

About VRML

The Virtual Reality Modeling Language (VRML) is a file format for describing interactive 3D objects and worlds. VRML is designed to be used on the Internet, intranets, and local client systems. VRML is also intended to be a universal interchange format for integrated 3D graphics and multimedia. VRML may be used in a variety of application areas such as engineering and scientific visualization, multimedia presentations, entertainment and educational titles, web pages, and shared virtual worlds.

Design Criteria

VRML has been designed to fulfill the following requirements:

Authorability

Enable the development of computer programs capable of creating, editing, and maintaining VRML files, as well as automatic translation programs for converting other commonly used 3D file formats into VRML files.

Composability

Provide the ability to use and combine dynamic 3D objects within a VRML world and thus allow re-usability.

Extensibility

Provide the ability to add new object types not explicitly defined in VRML.

Implementability

Capable of implementation on a wide range of systems.

Performance

Emphasize scalable, interactive performance on a wide variety of computing platforms.

Scalability

Enable arbitrarily large dynamic 3D worlds.

Characteristics of VRML

VRML is capable of representing static and animated dynamic 3D and multimedia objects with hyperlinks to other media such as text, sounds, movies, and images. VRML browsers, as well as authoring tools for the creation of VRML files, are widely available for many different platforms.

VRML supports an extensibility model that allows new dynamic 3D objects to be defined and a registration process that allows application communities to develop interoperable extensions to the base standard. There are mappings between VRML objects and commonly used 3D application programmer interface (API) features.

Scope

The VRML specification defines a file format that integrates 3D graphics and multimedia. Conceptually, each VRML file is a 3D time-based space that contains graphic and aural objects that can be dynamically modified through a variety of mechanisms. VRML defines a primary set of objects and mechanisms that encourage composition, encapsulation, and extension.

The semantics of VRML describe an abstract functional behaviour of time-based, interactive 3D, multimedia worlds. VRML does not define physical devices or any other implementation-dependent concepts (e.g., screen resolution and input devices). VRML is intended for a wide variety of devices and applications, and provides wide latitude in interpretation and implementation of the functionality. For example, VRML does not assume the existence of a mouse or 2D display device.

Each VRML file:

1. implicitly establishes a world coordinate space for all objects defined in the file, as well as all objects recursively included by the file;
2. explicitly defines and composes a set of 3D and multimedia objects;
3. can specify hyperlinks to other files and applications;
4. can define object behaviors.

An important characteristic of VRML files is the ability to compose files together through inclusion and to relate files together through hyperlinking. For example, consider the file earth.wrl which specifies a world that contains a sphere representing the earth. This file may also contain references to a variety of other VRML files representing cities on the earth (e.g., file paris.wrl). The enclosing file, earth.wrl, defines the coordinate system that all the cities reside in. Each city file defines the world coordinate system that the city resides in but that becomes a local coordinate system when contained by the earth file.

Hierarchical file inclusion enables the creation of arbitrarily large, dynamic worlds. Therefore, VRML ensures that each file is completely described by the objects and files contained within it and that the effects of each file are strictly scoped by the file and the spatial limits of the objects defined in the file. Otherwise, the accumulation of files into larger worlds would produce unscalable results (as each added world produces global effects on all other worlds). For example, light sources have the potential of global effect since light energy theoretically does not dissipate to zero. And, if the earth file contains 100 city files each containing 100 lights each affecting all objects in the world, the lighting calculations would quickly become intractable. Therefore, in order to prevent global effects, light source objects are scoped by either a maximum radius or by location within the file.

Another essential characteristic of VRML is that it is intended to be used in a distributed environment such as the World Wide Web. There are various objects and mechanisms built into the language that support multiple distributed files, including:

1. in-lining of other VRML files;
2. hyperlinking to other files;
3. using established Internet standards for other file formats;
4. defining a compact syntax.

See also: http://www.web3d.org/Specifications/VRML97/

Introduction to the VrmlPad Automation

The VrmlPad Automation provides application programming interface (API) that enables authors and developers to manipulate the program from external applications using the C, C++, Visual Basic®, Delphi, and scripts within HTML pages.

In addition, power users can now create add-ins, VBScript and JavaScript macros and execute them from within the VrmlPad environment, automating custom tasks. VrmlPad macros are procedures you write in the VBScript or JavaScript language, and add-ins are in-process ActiveX components (DLLs) you write in C, C++, Pascal or Visual Basic. Each add-in can add one or more commands, and each command can perform one or more tasks. The number of commands and tasks is up to you.

If your tasks are routine or repetitive, you can benefit from automating them. Through Automation (formerly OLE Automation), you can reduce the time spent on these tasks, and you can prevent errors that often result from performing the tasks manually. Also, you can add visual support for editing some VRML nodes in addition to text editing provided by VrmlPad.

With Automation, you perform tasks by manipulating VrmlPad and its components as objects. For example, you open, edit, and close a document by manipulating it as an object. Similarly, you get or edit a VRML node's properties by manipulating it as an object. Each VrmlPad object implements a dual interface through which you can manipulate the object.

You manipulate objects by using methods, properties, and events associated with the objects. Methods represent actions you take against the objects. Properties represent characteristics of the objects, such as their type or size. And, events represent conditions under which actions are taken against the objects.

Suppose you want to create a VRML file, containing WorldInfo node, quickly — by simply choosing a menu item. Start by writing a VBScript macro that will create the file. The macro looks like this:

```
Sub Create_New
Set doc = Window.Documents.Open
doc.Selection = "#Created by the Author" + vbCrLf
Set node = doc.RootNodes.Add("WorldInfo")
node.Fields("info").Add "Powered by VrmlPad"
doc.Saved = True
End Sub
```

This macro does the following:

- The Sub statement begins the macro, which has the name Create_New. The name is arbitrary, you can use whatever name you choose.
- The first line of the macro creates a new empty document, using the Open method of the Documents object.

- The second line of the macro adds a comment string and moves the caret to the next line, using the Selection property of the Document object.
- The next line of the macro adds an empty *WorldInfo* node, using the RootNodes collection of the Document object.
- The next line of the macro adds the *info* field to the node, using the Fields collection of the Node object, and sets field's value, using the Add method of the Field object.
- The last line of the macro marks the document as unmodified.
- The End Sub statement ends the macro.

After writing the macro, copy the macro file to the AddIns directory. Then, select the menu item Create New from the Tools menu to run the macro.

For more information about VrmlPad objects and writing add-ins and macros, see http://www.parallelgraphics.com/developer/products/vrmlpad/.

Text Editor

Overview: Text Editor

The VrmlPad environment includes an integrated text editor to manage, edit, and print source files. Most of the procedures for using the editor should seem familiar if you have used other Windows-based text editors. With the Text editor, you can:

- Set and customize syntax coloring for source VRML files.
- Use AutoComplete for quick entering a VRML keyword, node type, node name, field name, default field value or another syntax element.
- Perform advanced find and replace operations in a file, including using regular expressions.
- Use virtual spaces for advanced cursor positioning.
- Navigate through sections of code using the Go To dialog box.
- Use Bookmarks to mark frequently accessed lines in your source file.
- Customize the Text editor with save preferences, tabs, and indents.
- Modify the font style, size, and color.
- Select lines or multiple lines, copy and cut selection into clipboard.
- Split the Text editor window into two panes.
- Hide a body of huge or uninteresting nodes and prototype declarations.
- Use drag-and-drop editing within editor window, and between the Text editor and other applications.

Tip: While using the Text editor, in many instances you can click the right mouse button to display a shortcut menu of frequently used commands. The commands available depend on what the mouse pointer is pointing to.

Syntax Coloring

```
PROTO Message [
    field SFNode console NULL
    field MFString text []
    eventIn SFTime input
]{
    Script {
        field MFString text IS text
        field SFNode console IS console
        eventIn SFTime input IS input
        url "javascript:
        function input (val) {
            console.text = text;
            console.startTime = val;
        }"
    }
}
```

Syntax coloring uses different colors for various code elements, such as keywords, comments, strings, URLs, nodes, fields and field values. This coloring gives you visual cues about the structure and state of your code.

To change colors used for syntax coloring

1. From the **Tools** menu, choose **Options**.

2. Select the **Format** tab.
3. In the **Font** box, select the font you want.
4. The **Font** box displays the different fixed-pitch fonts installed on your system. The text sample in the **Sample** box changes to the font you select.
5. In the **Size** box, select the size to be used with the font you selected.

   The **Size** box displays the sizes available for the selected font. The text sample in the **Sample** box changes to the size you select.
6. In the **Colors** box, select the type of text you want to color.
7. In the **Background** box, select a background color.
8. In the **Foreground** box, select a foreground color.

   The **Background** and **Foreground** boxes display the standard colors, the **Custom** setting and the **Automatic** setting. The text sample displayed in the **Sample** box changes to the color you select.
   The behavior of the **Automatic** setting depends on the element selected. For colors that map to standard system elements (such as **Text** or **Selected Text**), the **Automatic** setting sets the element to the appropriate system color. For syntax coloring elements and other non-system defined colors, the **Automatic** setting indicates that the foreground color or background color from the parent category is to be used.
9. Click **OK**.

To disable syntax coloring

1. From the Tools menu, choose Options.
2. Select the **Editor** tab.
3. In the **Options** box, unselect **Highlight language syntax** checkmark.
4. Click **OK**.

AutoComplete

The Complete command (Ctrl+Space) opens a dropdown list box in the Text editor that contains the VRML identifiers (keywords, PROTO, node and field names, fields types) appropriate to the grammatical context at the current caret position. To have the list box automatically open as you type your code, select Auto list identifiers on the Editor tab in the Options dialog box.

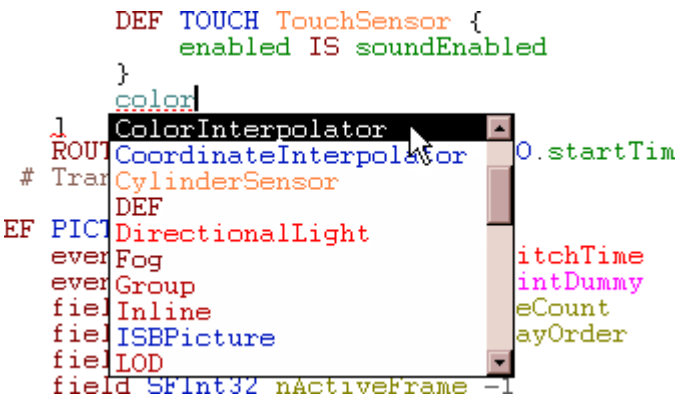
You can find the identifier you want in the list box by:

- Typing the name.

  As you type, the identifier that matches the characters you type is selected and moves to the top of the list.
- Using the up and down arrow keys to move up and down in the list.
- Scrolling through the list and selecting the identifier you want.

You can insert the identifier into your code by:

- Clicking the identifier.
- Selecting the identifier and pressing SPACE or ENTER to insert the selection.



You can also type Ctrl+Space after a node or PROTO identifier. The command adds braces after identifier, inserts empty line and sets there the caret.

After a field declaration or definition the command inserts default field value.

Error Processing

As you type, VrmlPad can automatically check your document and underline syntax and semantic errors or possible warnings. There are three types of errors indicated on the status bar at the bottom of the VrmlPad window:

- Syntax errors.

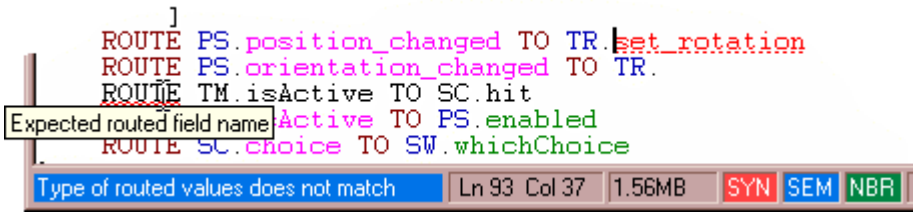
  Wrong construction underlines by red wavy line and indicates on the status bar by a red pane labeled **SYN**.

- Semantic errors and warnings. Missing, duplicated or type-mismatched identifiers.

  Identifier underlines by red dash line and indicates on the status bar by a blue pane labeled **SEM**.

- Nonmatching braces, square brackets and double quotes.

  Status bar indicates this error by a green pane labeled **NBR**.



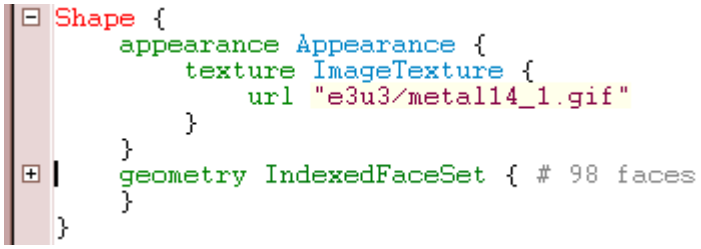
To get explanation for an error:

- Rest the mouse pointer over the wrong construction for a moment and read a ScreenTip for the error.
- Double-click the highlighted pane on the status bar. The caret will move to the first error of that type.

To quickly move to the next error, choose **Next Error** from the **View** menu, or type F4 (to go to the previous error, type Shift+F4).

Node Folding

Node folding lets you hide a body of huge or uninteresting nodes and prototype declarations. The lines of code inside these nodes can be expanded or collapsed using a [+]/[-] icon on the selection margin at the left side of Text editor.



To expand or collapse a node fold

- Click the **+** icon to expand the fold or **–** icon to collapse.
- Or set the caret at the folded lines and from the **Edit / Fold** submenu, choose **Expand** or **Collapse** or press **Alt+Down Arrow** or **Alt+Up Arrow**.
- Or right-click the fold sign and from the pull-down menu choose **Toggle**.

To expand or collapse all node folds

- From the **Edit / Fold** submenu, choose **Expand All** or **Collapse All**, or
- Right-click any fold sign and from the pull-down menu choose **Expand All** or **Collapse All**.

By default, a number of lines that can be folded should be at least 20 lines.

To change the minimum number of lines to fold

1. From the **Tools** menu, choose **Options**.
2. Select the **Node Folds** tab.
3. In the **Minimum length of node to fold** box, type the number of lines.

   Tip: Type **0** lines to fold all nodes and prototype declarations.
4. Click **OK**.

You can define a set of node types that should be hidden normally. All nodes of such uninteresting types can be collapsed manually or automatically (on scene load or text insertion).

To define a set of uninteresting nodes

1. From the **Tools** menu, choose **Options**.
2. Select the **Node Folds** tab.
3. Using **Add >>** and **<< Remove** buttons, modify a list of nodes in the **Uninteresting nodes** box.
4. To automatic collapse uninteresting nodes, select **Automatically collapse uninteresting nodes on load and inserts** checkbox.

To disable node folding

1. From the **Tools** menu, choose **Options**.
2. Select the **Editor** tab.
3. In the **Options** box, unselect **Allow node folding** checkmark.
4. Click **OK**.

Finding Text

With the advanced find and replace capabilities of the Text editor, you can search for literal text strings or use regular expressions to find words or characters. A regular expression is a search string that uses special characters to match a text pattern in a file. You can use regular expressions, including tagged regular expressions, with both the Find and Replace commands.

With the Find and Replace commands, you can:

- Find text in a document.
- Replace text in a whole document.
- Replace text in a part of a document.
- Use regular expressions.

To find a text string

1. Move the insertion point to where you want to begin your search.

   The editor uses the location of the insertion point to select a default search string.
2. From the **Edit** menu, choose **Find** or type Ctrl+F.
3. In the **Find what** box, type the search text or a regular expression.

   Tip: Select the menu button to the right of the box to display a list of regular search expressions. When you select an expression from this list, the expression is substituted as the search text in the Find what box. If you do use regular expressions, be sure the Regular expression check box is selected. You can also use the drop-down list to select from a list of up to 16 previous search strings.
4. Select any of the **Find** options.
5. Start the search by clicking the **Find Next** or **Mark All** buttons.

To start a find without using the Find dialog box

- Type Ctrl+F3 to search a word under the caret position.
- To continue your search, type the **Find Next** (F3) or **Find Previous** (Shift+F3) buttons.

To replace text

1. Move the insertion point to where you want to begin your search.
2. The editor uses the location of the insertion point to select a default search string.
3. From the **Edit** menu, choose **Replace**.
4. In the **Find what** text box, type the search text or a regular expression.

   Tip: Click the button to the right of the box to display a list of regular search expressions. When you select an expression from this list, the expression is substituted as the search text in the Find what text box. If you do use regular expressions, be sure the Regular expression check box is selected. You can also use the drop-down list to select from up to 16 previous search strings.
5. In the **Replace with text** box, type the replacement text.

   Tip: Click the button to the right of the box to display a list of tagged expressions you can use as replacement text.
6. Select any of the remaining **Find** options.
7. Start the search by clicking the **Find Next**, **Replace**, or **Replace All** buttons.

Regular Expressions

Regular Expression	Description
.(Period.)	Any single character
[ ]	Any one of the characters contained in the brackets, or any of an ASCII range of characters separated by a hyphen (-). For example, b[aeiou]d matches bad, bed, bid, bod, and bud, and r[eo]+d matches red, rod, reed, and rood, but not reod or roed. x[0-9] matches x0, x1, x2, and so on. If the first character in the brackets is a caret (^), then the regular expression matches any characters except those in the brackets.
^	The beginning of a line.
$	The end of a line.
( )	Indicates a tagged expression to retain for replacement purposes. If the expression in the Find what text box is (PROTO)OldName, and the expression in the Replace With box is \1NewName, all selected occurrences of PROTO OldName are replaced with PROTO NewName. Each occurrence of a tagged expression is numbered according to its order in the Find what text box, and its replacement expression is \n, where 1 corresponds to the first tagged expression, 2 to the second, and so on. You can have up to nine tagged expressions.
c|c	Any one of the characters separated by the alternation symbol (|). For example, (j|u)+fruit finds jfruit, jjfruit, ufruit, ujfruit, uufruit, and so on.
*	None or more of the preceding characters or expressions. For example, ba*c matches bc, bac, baac, baaac, and so on.
+	At least one or more of the preceding characters or expressions. For example, ba+c matches bac, baac, baaac, but not bc.
?	None or one of the preceding characters or expressions. For example, ba?c matches bc, and bac.
( )	Any sequence of characters between the escaped braces. For example, (ju)+fruit finds jufruit, jujufruit, jujujufruit, and so on. Note that it will not find jfruit, ufruit, or ujfruit, because the sequence ju is not in any of those strings.
[^]	Any character except those following the caret (^) character in the brackets, or any of an ASCII range of characters separated by a hyphen (-). For example, x[^0-9] matches xa, xb, xc, and so on, but not x0, x1, x2, and so on.
\a	Any single alphanumeric character [a – zA – Z0 – 9].
\w+	Any white-space characters. The \w+ finds tabs and spaces.
\c	Any single alphabetic character [a – zA – Z].
\d	Any decimal digit [0 – 9].
\v	Any VRML identifier.
\s	Any quoted string ("[^"]*").
\	Removes the pattern match characteristic in the Find what text box from the special characters listed above. For example, 100$ matches 100 at the end of a line, but 100\$ matches the character string 100$ anywhere on a line.

File Navigation

The Text editor provides a variety of methods to move around in a source file. In addition to the standard Windows navigation mechanisms, the Text editor includes an assortment of commands that enable you to move to almost any location in a file. Further, in addition to comprehensive GoTo and Bookmarks commands, the Text editor includes several advanced navigation features, such as virtual space.

About Virtual Space

All editors support moving the cursor by one character position. The most common difference among text editors is whether you can move the cursor into a location that does not currently contain text. For example, if your cursor is on column 20 and there is no text on the line below the current line, moving the cursor down can do one of two things: Either the cursor moves to column 1 — because there is no text on the line below — or the cursor moves to column 20 of the next line. This behavior is called virtual space.

With the Text editor, you can treat text selection and space insertion in two ways. When you select the Virtual Spaces option, spaces are inserted between the end of the line and the insertion point before new characters are added to the line. When you clear the Virtual Spaces option, the Text editor behaves like Microsoft Word for Windows — the insertion point is set to the end of the line.

To enable virtual space

1. From the **Tools** menu, choose **Options**.
2. Select the **Editor** tab.
3. In the **Options** box, select **Enable virtual space** checkmark.
4. Click **OK**.

About Go To

The Go To dialog box allows you to jump quickly to several different items in a file, including:

- Lines (type the line number)
- Lines in inline scripts (type the line number from the beginning of the script and scripts node name)
- PROTO, node and field declarations (type or select the name of a PROTO, node, or field to go to where it is defined)
- PROTO, node and field references (type or select the name of a PROTO, node, or field to go to where it is referenced)
- ROUTEs (type or select the known node and field names to go to where it is routed)

To use the Go To dialog box

1. From the **Edit** menu, choose **Go To**.
2. In the **Go To what** box, select the type of item you want.
3. Enter any additional information required.
4. Click one of the navigation buttons: **Go To**, **Previous**, or **Next**.

Tip: To quickly jump to definition of an identifier (PROTO, node, or field name) under the caret position, type Ctrl+F11. To jump to an identifier reference, type F11.

About Bookmarks

You can set bookmarks to mark frequently accessed lines in your source file. Once a bookmark is set (Ctrl+F2), you can use menu or keyboard commands (F2 and Shift+F2) to move to it. You can remove a bookmark when you no longer need it (Ctrl+F2 again).

Using Drag-and-Drop Editing

Drag-and-drop editing is the easiest way to move or copy a selection of text within a file, between files, or between applications. The text you drop remains selected, which makes it easy to copy a chunk of text into several places.

To move text using drag-and-drop editing

1. Select the text you want to move.
2. Drag the selected text to the new location.

   Note: You can also use the right mouse button for drag-and-drop editing. Select the text you want, and then use the right mouse button to drag the text to a new location. A shortcut menu appears, asking if you want to move or copy the selected text.

   Tip: At any time during a drag-and-drop procedure, you can click the other mouse button to cancel the operation.

To copy text using drag-and-drop editing

1. Select the text you want to copy.
2. While holding down the CTRL key, drag the selected text to the new location.

To move or copy text to another document

1. Select the text you want to move or copy.
2. Drag the selected text to the document bar and hold it over the document tab.
3. After the target document selected drag the text to the new location.

Editor Commands and Keystrokes

You can access Text editor commands in many ways: from the menus, from the toolbar, and from the shortcut menu. There are many Text editor commands, but not all of them appear on the menus and toolbars. The Selection menu (available from the Edit menu) contains some of the most useful edit commands, including Format, Indent, Unindent, Tabify, Untabify, Make Uppercase, Make Lowercase, and Comment.

Use these key combinations in the Text editor window:

Press	To
F1	Get context-sensitive Help on node type under the caret position.
F2	Moves to the line containing the next bookmark.
Shift+F2	Moves to the line containing the previous bookmark.

Ctrl+F2	Toggles a bookmark for the current line on and off.
F3	Finds the next occurrence of the specified text.
Shift+F3	Finds the previous occurrence of the specified text.
Ctrl+F3	Finds the next occurrence of the selected text.
Ctrl+Shift+F3	Finds the previous occurrence of the selected text.
Alt+F3	Finds the specified text.
F4	Moves to the line containing the next error or warning.
Shift+F4	Moves to the line containing the previous error or warning.
Ctrl+F4	Closes the active document.
F5	Preview the document, using default Web browser and VRML viewer.
F11	Displays an identifier definition.
Shift+F11	Displays an identifier reference.
Ctrl+A	Selects the entire document.
Ctrl+C	Copies the selection to the Clipboard.
Ctrl+Shift+C	Comment or uncomment the selected lines.
Ctrl+E	Marks a node or a field at the caret position as a route source.
Ctrl+F	Activates the Find tool.
Ctrl+Shift+F	Formats the selection using the smart indent settings.
Ctrl+G	Moves to a specified location.
Ctrl+L	Deletes the selected lines and puts them on the Clipboard.
Ctrl+R	Marks a node or a field at the caret position as a route target and opens **Add Route** dialog box.
Alt+S	Synchronize the caret position with a selection in the Scene Tree.
Ctrl+Shift+T	Replaces spaces with tabs in the selection.
Ctrl+Alt+T	Shows or hides tab characters.
Ctrl+U	Makes the selection all lowercase
Ctrl+Shift+U	Makes the selection all uppercase.
Ctrl+V	Inserts the Clipboard contents at the insertion point.
Ctrl+X	Cuts the selection and moves it to the Clipboard.
Ctrl+Y	Redoes the previously undone action.
Ctrl+Z	Undoes the last action.
Ctrl+]	Finds the matching brace.
Ctrl+Shift+]	Extends the selection to the matching brace.
Ctrl+Space	Opens a dropdown list box that contains the available identifiers.
Ctrl+Shift+Space	Replaces tabs with spaces in the selection.
Tab	Indents the selection.
Shift+Tab	Unindents the selection.
Ctrl+Tab	Activates next document.
Ctrl+Shift+Tab	Activates previous document.
Backspace	Deletes the selection or, if there is no selection, the character to the left of the cursor.
Ctrl+Backspace	Deletes a word to the left.
Del	Deletes the selection.
Ctrl+Del	Deletes a word to the right.
Left Arrow	Moves the cursor one character to the left.

Shift+Left Arrow	Extends the selection one character to the left.
Ctrl+Left Arrow	Moves back one word.
Ctrl+Shift+Left Arrow	Extends the selection back one word.
Right Arrow	Moves the cursor one character to the right.
Shift+Right Arrow	Extends the selection one character to the right.
Ctrl+Right Arrow	Moves forward one word.
Ctrl+Shift+Right Arrow	Extends the selection forward one word.
Up Arrow	Moves the cursor up one line.
Shift+Up Arrow	Extends the selection up one line.
Ctrl+Up Arrow	Scrolls the file contents down one line.
Alt+Up Arrow	Collapse the node fold at the cursor
Down Arrow	Moves the cursor down one line.
Shift+Down Arrow	Extends the selection down one line.
Ctrl+Down Arrow	Scrolls the file contents up one line.
Alt+Down Arrow	Expand the node fold at the cursor
Home	Moves to either the start of the current line or the start of the text on that line.
Shift+Home	Extends the selection to either the start of the current line or the start of the text on that line.
Ctrl+Home	Moves to the beginning of the file.
Ctrl+Shift+Home	Extends the selection to the beginning of the file.
End	Moves to the end of the current line.
Shift+End	Extends the selection to the end of the current line.
Ctrl+End	Moves to the end of the document.
Ctrl+Shift+End	Extends the selection to the end of the document.
Page Down	Moves the cursor down one page.
Page Up	Moves the cursor up one page.
Shift+Page Down	Extends the selection down one page.
Shift+Page Up	Extends the selection up one page.
Ins	Toggles between inserting and replacing text.

Scene Tree

Overview: Scene Tree

The Scene Tree displays VRML hierarchy of your scene and lets you:

- Browse and edit the VRML hierarchy of your scene.
- Edit node, PROTO, and field names.
- Make precise selections within the Text Editor.

For each node, the Scene Tree displays:

- An icon indicating the node type (for Transform , for Shape , and so on).

  Each node type has a particular icon that is associated with it. Clones (multiple instances) of a node are indicated with the link mark above the node icon.
- The name of the node type (Transform for instance).
- The name (if any) that you've given to the node.

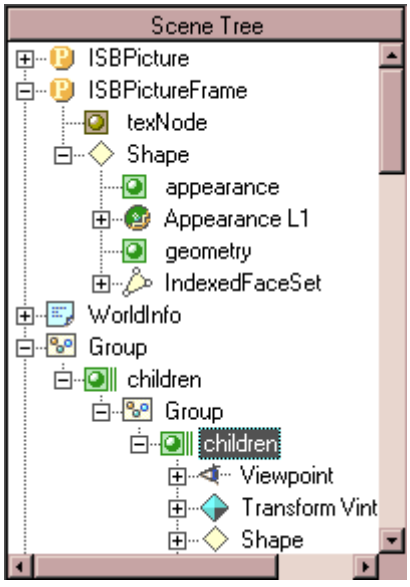
For each PROTO interface declaration, the Scene Tree displays:

- An PROTO icon .
- The name of the PROTO.

For each field declaration or definition, the Scene Tree displays:

- An icon indicating the field type and category (for exposedField SFNode , for eventIn SFFloat , and so on).

  ISed fields are indicated with the link mark above the field icon.
- The name of the field (children for instance).



To show only nodes and PROTOs in the Scene Tree, select Show nodes only on the Tree tab in the Options dialog box.

Scene Tree Navigation

You can see various levels of the hierarchy contained under nodes and PROTOs by clicking the node icon. Opening a node refers to expanding the hierarchy beneath it. Closing a node refers to collapsing the hierarchy beneath it. You can also click a MFNode field icon to toggle it open and see its values.

To open or close a node, PROTO, or field:

- Click the + icon to open the item.
- Click the – icon to close the item.
- Type the * button to open all subitems of the selected item.

To jump to the definition of the selected item (PROTO-instance, nodes clone, or field):

- Choose **Go To Definition** from the pull-down menu (right-click the item to see the menu).
- Or press Ctrl+F11.

To jump to the reference of the selected item (PROTO-declaration, node, or field declaration):

- Choose **Go To Reference** from the pull-down menu.
- Or press F11.

You can synchronize current selection in the Scene Tree with the caret position in the Text Editor in both directions.

To set the caret position in the Text editor at a node, PROTO or field:

- Double-click the item.
- Or select the item and press ENTER.
- Or choose **Go To** from the pull-down menu.

To set the selection in the Scene Tree to the node, PROTO or field near the caret position:

1. Activate Text editor window.
2. Choose **Synchronize** from the **View** menu (or type Alt+S).

To have the Scene Tree automatically synchronize selection as you type your code, select Automatic synchronize context on the Tree tab in the Options dialog box.

Moving, Copying, and Cloning Nodes within the Scene Tree

The top-level nodes in a VRML file are in no particular order; so you can reorder them to suit your tastes without changing what your world looks like. You can also move a node (other than a top-level node) or field declaration/definition to reorganize the structure of your world.

To move a node (field) in the scene hierarchy:

1. Select the node (field) you want to move.
2. Click and drag the node (field) icon. When the pointer is over a place where the item can legally be inserted, two red arrows appear; if you release the mouse button at that spot, the node (field) is inserted there.

Note: When the pointer is over a closed grouping node or a MFNode field, rest the mouse pointer over the item for a second and the item will open.

To copy a node in the scene hierarchy:

1. Select the node you want to copy.
2. While holding down the CTRL key, drag the selected node to the new location.

To clone a node in the scene hierarchy:

1. Select the node you want to clone.
2. While holding down the ALT key, drag the selected node to the new location below current selection.

Note: If the dragged node is unnamed, Scene Tree prompts you for the node name.

If you use your right mouse button to drag, a menu appears with the available options.

Naming Nodes in the Scene Tree

Using the Scene Tree, you can rename a node, PROTO declaration or field declaration with appropriate renaming of all its instances (clones).

To edit the node name using the Scene Tree:

1. Select the node you want to rename.
2. Press F2 or right-click over the node and choose **Rename**.
3. Type a name in the text box and press Enter or press Esc if you want to leave the text box unchanged.

   Note: VRML node names are not allowed to contain spaces, single or double quotation marks, pound signs (#), commas, periods, square brackets, backslashes, or curly braces. Also, they can't begin with a numerical digit or the plus (+) or minus (-) characters.

To clear the node name:

1. Select the node you want to unname.
2. Press F2 or right-click over the node and choose **Rename**.
3. Clear the text box and press Enter.

   Note: If the node name is in use, the Scene Tree warns you about it.

To rename a PROTO (with all PROTO-instances of this PROTO):

1. Select the PROTO declaration you want to rename.
2. Press F2 or right-click over the PROTO and choose **Rename**.
3. Type a name in the text box and press Enter or press Esc if you want to leave the text box unchanged.

To rename a field declaration (with all references to this field):

1. Select the field declaration you want to rename.
2. Press F2 or right-click over the field and choose **Rename**.
3. Type a name in the text box and press Enter or press Esc if you want to leave the text box unchanged.

Note: To prevent renaming field references inside the inlined vrml-script body, select Disable renaming in a script body check box on the Tree tab in the Options dialog box.

To remove a node, PROTO or field declaration

1. Select the item you want to remove.
2. Press Del or right-click over the item and choose **Delete**.

   Note: To remove also all references to this item (node and PROTO instances, routes, PROTO field implementations) and items within this item, press Ctrl+Del key or choose Delete from the pull-down menu while holding down the CTRL key.

Routing Map

Overview: Routing Map

The Routing Map displays routes of your scene and lets you browse and edit connections between events and exposed fields in your scene.

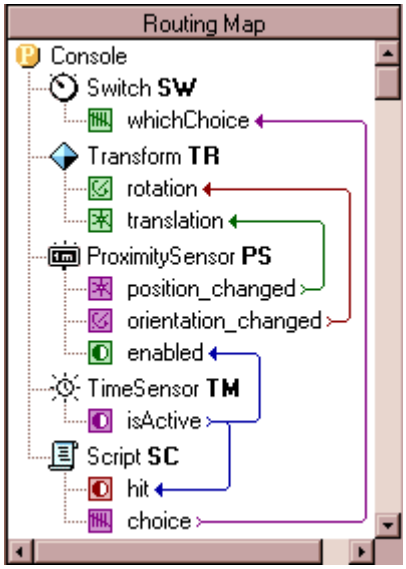
For each route, the Routing Map displays:

- An icon and a label indicating the PROTO-context (namespace) of the route, if available.
- Icons and labels indicating the source and target nodes. Some icons may be shared between two or more routes.

  By default, Routing Map displays only node DEF-names. To show also typenames of nodes, select Show node typenames on the Tree tab in the Options dialog box.
- Icons and labels indicating the source and target fields.

  To show also icons indicating actual field owners for ISed prototype interface fields, select Show actual field owners on the Tree tab in the Options dialog box.
- A colored line indicating the route itself.

  To change a color corresponding to some route type, select the route type and desired color in the Route colors box on the Tree tab in the Options dialog box.



Managing Routes

To set the caret position in the Text editor at a route:

- Double-click the route line. To move to the route source or target node or field declaration, double-click the appropriate icon or label.
- Choose **Go To** from the pull-down menu (right-click the route line to see the menu). To move to the route source or target node (or scripts field declaration), choose **Go To Source** or **Go To Target**.

To set the selection in the Routing Map to the route near the caret position:

1. Activate Text editor window.
2. Choose **Synchronize** from the **View** menu (or type Alt+S).

   If the caret position is near a node, field definition or event declaration, the Synchronize command enumerates all routes connecting this node or field with some other fields.

To select next or previous route:

- Press Down or Up. Left and Right buttons scrolls Routing Map to show both ends of a long selected route. Typing Ctrl+Page Down and Ctrl+Page Up scrolls Routing Map one page down or up without changing the selection.

To remove a route:

1. Select the route you want to remove.
2. Press Del or right-click over the route line and choose **Delete Route**.

To redirect a route:

1. Select the route you want to change.
2. Right-click over the route line and choose **Add Route** (or type Ctrl+R).
3. Select a new source or/and target node or event name and click OK.

To add a new route:

1. Activate Text editor or Scene Tree window.
2. Select in Scene Tree or move the caret at the source node, scripts eventOut or exposedField.
3. Choose **Start Route** from the **Edit** menu (or type Ctrl+E) to mark the node or the field as a source for the new route.
4. Select in Scene Tree or move the caret at the target node, scripts eventIn or exposedField.
5. Choose **Add Route** from the **Edit** menu (or type Ctrl+R) to open **Add Route** dialog box.
6. Enter, if necessary, unique names for the source or/and target nodes and fill out missing field names.
7. Check **Insert the route at the caret position** if you want to place newly created route near the caret position in Text editor. Otherwise, the new route will be inserted at the end of the file or the prototype declaration.
8. Click **OK**.

Resource View

Overview: Resource View

The Resource View displays dependencies (resources) of your scene and lets you browse and edit links to the resources.

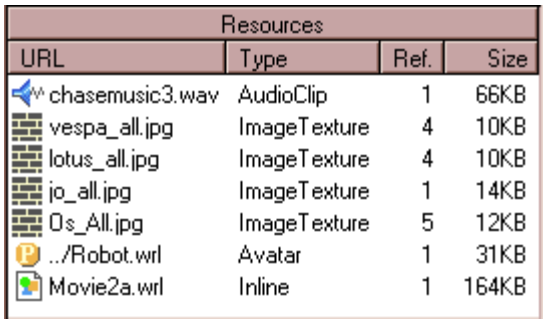
For each resource, the Resource View displays:

- An icon indicating the resource type (for ImageTexture, for AudioClips, for custom resources, and so on).

  To determine type of the resource, the Resource View finds the final declaration (unwinding all IS statements) of the field or exposed field of type MFString, containing the URL to the resource as a first string. The type of standard node or PROTO, that contain this declaration, becoming a type of the resource.

  Remote resources (beginning with 'file:' protocol or with no explicit protocol) are indicated with the link mark above the resource icon.

  Missing local resources are indicated with the cross mark above the resource icon.
- The URL of the resource.
- The name of the resource type (ImageTexture for instance).
- The reference count to the resource (number of links to this resource).
- The size of the local resource.



Managing Resources

You can easily enumerate all links (URLs) to the resource.

To set the caret position in the Text editor at a resource URL:

- Double-click the item. To move to the next link, double-click the item again.
- Choose **Select** from the pull-down menu (right-click the item to see the menu).

The list of items in the Resource View may be sorted by URL, by type, by reference count or by file size.

To sort list of resources:

- Right-click an empty space or the header of the Resource View. The **Sort by** entries in the pull-down menu let you specify the sort order.

- Or click on the appropriate portion of the list box header. For example, click on the word Size in the header to sort by size. To reverse the sort order, click it again.

Using the Resource View, you can change all links to some resource to the another resource.

To redirect links to a resource:

1. Select the resource you want to relink.
2. Press F2 or right-click over the resource and choose **Rename**.
3. Type a name in the text box and press Enter or press Esc if you want to leave the text box unchanged.

To browse for a local resource:

1. Right-click over the resource and choose **Browse**.
2. Select a resource file you want to link to.
3. Check **Store absolute path** if you want to store absolute URL to the resource. Otherwise, relative path to the resource will be inserted, if possible.
4. Click **OK**.

To convert absolute path to a local resource to relative and vice versa:

1. Right-click over the resource and choose **Browse**.
2. Check or uncheck **Store absolute path**.
3. Click **OK**.

If the resource has file associations, you can open it for previewing or editing using the Resource View.

To refresh list of resources after external files modification:

- Right-click an empty space or the header of the Resource View. Choose **Refresh** from the pull-down menu.

To open a resource using associated application:

- Right-click over the resource and choose **Open** or **Edit**.

Using Drag-and-Drop in the Resource View

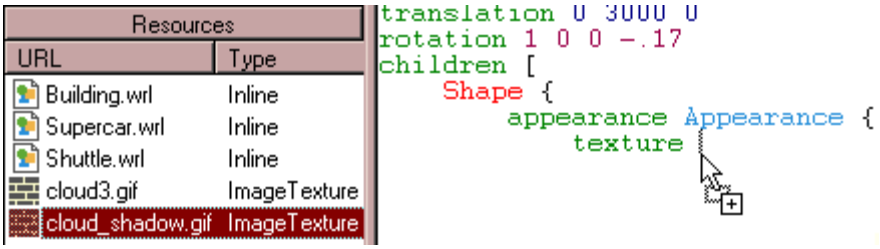
You can easily create a new link to resource, using drag-and-drop in the Resource View. You can also move or copy a resource file to an another application (Windows Explorer, for instance).

To create a new link to a resource:

1. Select the resource you want to link to.
2. Drag the selected resource to the location in the Text editor.

   When the pointer is over a place where the resource link can legally be inserted, dimmed caret appear; if you release the mouse button at that spot, a new link to the resource is inserted there.

   Depending on the place you dropping the resource, only URL string may be inserted, or also a node of the resource type with the url field defined.



To copy resource file to an another application:

1. Select the resource you want to copy.
2. Drag the selected resource to the application.

   If you use your right mouse button to drag, a menu appears with the available options.

To open VRML resource in VrmlPad:

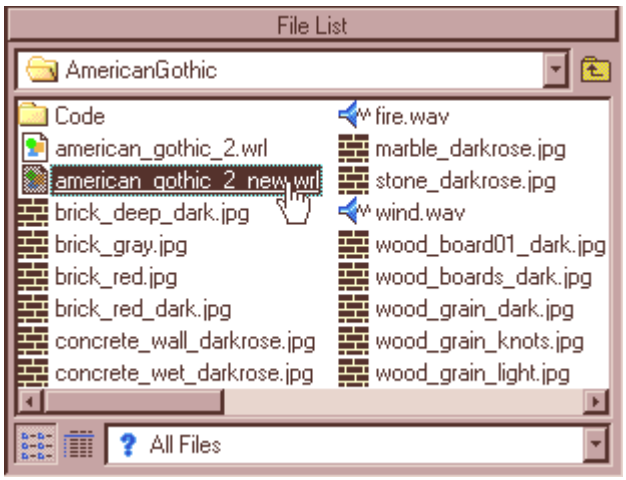
1. Select the VRML resource you want to open.
2. Drag the selected resource to the title bar of VrmlPad.

   If you drop a remote resource, VrmlPad automatically downloads it from the Net.

File List

Overview: File List

The File List displays drives, folders, and files accessible from your PC. The right-click menu contains file handling commands and you can open Windows Explorer directly to create folders, run searches, etc.



To open VRML file in VrmlPad:

1. Navigate to the drive and directory you want to access.
2. Double-click on a folder to display its files.
3. Double-click on a file to open it in the editor.

   Tip: To have the File List use single-click to open a file, select Single click to open a file (point to select) on the File List tab in the Options dialog box.

   Tip: To open a file in a new instance of VrmlPad, hold Shift key while opening the file.

You can set the file list to show specific file types. The filter is a global setting for all directories.

To filter the file list:

1. Open combo box at the bottom of the File List.
2. Select a file type from the list.

   The file list refreshes to apply the filter.

Using Drag-and-Drop in the File List

Drag-and-drop in the File List works like in Windows Explorer - you can drag files from and to the File List using left and right mouse buttons and CTRL or ALT keyboard modifiers.

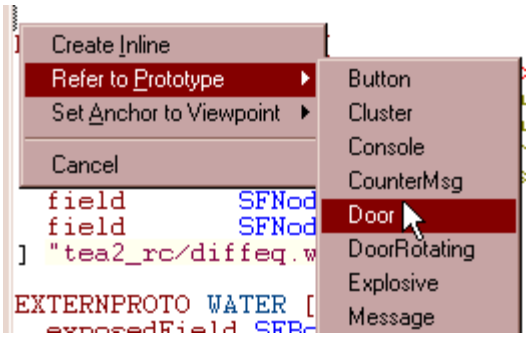
In addition, you can insert links to files in the Text editor, using drag-and-drop like in the Resource View**.**

To create a link to a file in the Text editor:

1. Select the file you want to link to.
2. Drag the selected file to the location in the Text editor.

When the pointer is over a place where the resource link can legally be inserted, dimmed caret appear; if you release the mouse button at that spot, a new link to the resource is inserted there.

Depending on the place you dropping the resource, only URL string may be inserted, or also a node of the resource type with the url field defined. In the case of VRML files, a pull-down menu appears; you can choose between an Inline, Anchor or EXTERNPROTO declaration.



Script Debugger

Overview: Script Debugger

VrmlPad incorporates an integrated script debugger to help locate bugs in interactive VRML scenes.

Debugging is the process of correcting or modifying the inline vrmlscript code in your VRML scene so that your scene can run smoothly, act as you expect, and be easy to maintained. To this end, VrmlPad provides a variety of tools to help with the varied tasks of tracking down errors in the code.

The debugger interface provides wrapper for the Cortona VRML client, special menus, windows, dialog boxes, and spreadsheet fields. Drag-and-drop functionality is available for moving debug information between components. Occasionally the debugger is paused in break mode, meaning the debugger is waiting for user input after completing a debugging command (like break at breakpoint, step into/over/out/to cursor, break after Break command or Restart).

Edit Time, Run Time, and Break Mode

To test and debug a VRML scene, you need to understand which of three modes you are in at any given time. You use VrmlPad at edit time to create a VRML scene, and at run time to run it. This chapter introduces *break mode*, which suspends the execution of vrmlscripts so you can examine and alter data. The VrmlPad title bar always shows you the current mode.

The characteristics of the three modes are listed in the following table.

Mode	Description
Edit time	Most of the work of creating a VRML scene is done at edit time. You cannot execute code or use debugging tools, except for setting breakpoints. From the Debug menu, choose Go, or click the Go button to switch to run time. If your VRML scene contains vrmlscript code that executes when the scene starts, choose Step Into from the Debug menu (or press F8) to place the VRML viewer in break mode at the first executable script statement.
Run time	When a VRML viewer takes control, you interact with the VRML scene the same way a user would. You can view code, but you cannot change it. From the Debug menu, choose Stop Debugging, or close the VRML viewer window to switch back to edit time.
Break mode	From the Debug menu, choose Break or Break in Script, click the Break button, or press CTRL+BREAK to switch to break mode. Execution is suspended while running the VRML viewer. You can view code, examine or modify data, restart the VRML viewer, end execution, or continue execution from the same point. You can set breakpoints at edit time, but other debugging tools work only in break mode.

At edit time, you can change the code of a VRML scene, but you cannot see how your changes affect the way the scene runs. At run time, you can watch how the scene behaves, but you cannot directly change the code.

Break mode halts the operation of a VRML client and gives you a snapshot of its condition at any moment. Variable and property settings are preserved, so you can analyze the current state of the VRML scene and enter changes that affect how the scene runs. When the VRML viewer is in break mode, you can:

- Determine which active procedures have been called.
- Watch the values of variables, properties, and statements.
- Change the values of variables and properties.
- View or control which statement the VRML client will run next.
- Run vrmlscript statements immediately.

Entering Break Mode at a Problem Statement

When debugging, you may want the VRML viewer to halt at the place in the code where you think the problem might have started. This is one reason VrmlPad provides breakpoints. A breakpoint defines a statement or set of conditions at which VrmlPad automatically stops execution and puts the VRML viewer in break mode without running the statement containing the breakpoint.

You can enter break mode manually if you do any of the following while the VRML viewer is running:

- Press CTRL+BREAK.
- Choose Break from the Debug menu.
- Click the Break button on the toolbar above Debugger windows.

It's possible to break execution when the VRML viewer is idle (when it is between processing of events). When this happens, execution does not stop at a specific line, but VrmlPad switches to break mode anyway. To break execution at a statement of vrmlscript code, choose Break in Script from the Debug menu.

You can also enter break mode automatically when execution reaches a line with a breakpoint or a statement generates a run-time error. A breakpoint halts the VRML viewer just before executing the line that contains the breakpoint. If you want to observe what happens when the line with the breakpoint executes, you must execute at least one more statement. To do this, use Step Into, Step Over, or Step Out. If necessary, you can also skip over statements using Run to Cursor. After stepping through each statement, you can see its effect by looking the debugger windows.

Step Mode	Description
Step Into	Execute the current statement and break at the next line, even if it's in another procedure. When you use Step Into to step through code one statement at a time, VrmlPad temporarily switches to run time, executes the current statement, and advances to the next statement. Then it switches back to break mode.
Step Over	Execute the entire procedure called by the current line and break at the line following the current line. Step Over is identical to Step Into, except when the current statement contains a call to a procedure or a single-line loop. Unlike Step Into, which steps into the called procedure, Step Over executes it as a unit and then steps to the next statement in the current procedure.
Step Out	Execute the remainder of the current procedure and break at the statement following the one that called the procedure.

When the VRML viewer is in break mode, you can use the Run to Cursor command to select a statement further down in your code where you want execution to stop. This lets you "step over" uninteresting sections of code, such as large loops.

Note: When you are trying to isolate a problem, remember that a statement might be indirectly at fault because it assigns an incorrect value to a variable. To examine the values of variables and properties while in break mode, use the Variables window, Quick Watch, or the Immediate window.

Debugger commands

Commands for debugging can be found on the Debug menu, the View menu, the Debug Windows toolbar and the main toolbar.

To start debugging, choose the Go, Run Embedded, Step Into or Run To Cursor command from the Debug menu. The following table lists the Debug menu commands that may appear and their actions.

Go

Runs the Cortona VRML client until a breakpoint is reached.
Keyboard shortcut: Shift+F5.

Run Embedded

Starts Microsoft Internet Explorer and runs specified HTML page (local or remote) with an embedded (through tags <OBJECT> or <EMBED>) Cortona VRML control. Executes scripts until a breakpoint is reached or an user stops or breaks the execution.

Step Into

Runs the Cortona VRML client and enters break mode before the first line of vrmlscript code is executed.
Keyboard shortcut: F8.

Run to Cursor

Runs the Cortona VRML client as far as the line that contains the insertion point. This is equivalent to setting a temporary breakpoint at the location of the insertion point.
Keyboard shortcut: Ctrl+F8.

When you begin debugging, the Debug Windows appear and you can control the program execution using the commands listed in the following table.

Go

Executes code from the current statement until a breakpoint is reached or an user stops or breaks the execution.
Keyboard shortcut: Shift+F5.

Restart

Restarts the Cortona VRML client and discards the current values of all variables (breakpoints and watch expressions still apply).
Keyboard shortcut: Ctrl+F5.

Break

Stops execution of the VRML client while it's running and switches to break mode. Any statement being executed when you choose this command is displayed in the Text editor with ➪ in the left margin if you have checked Show selection margin in the Editor tab of the Options dialog box. If the VRML client is waiting for events in the idle loop (no vrmlscript statement is being executed), no statement is highlighted until an event occurs.

Keyboard shortcut: Ctrl+Break.
Note: Editing changes made in the break mode may require you to restart the VRML client for the changes to take effect.

Break in Script

Runs the VRML client until a statement of vrmlscript code is executed. Then stops execution and switches to break mode.

Stop Debugging

Stops running the VRML client and returns to a normal editing session.
Keyboard shortcut: Alt+F5.

Step Into

Single-steps through instructions in the code, and enters each function call that is encountered.
Keyboard shortcut: F8.

Step Over

Single-steps through instructions in the code. If this command is used when you reach a function call or single-line loop, the function is executed without stepping through the function or loop instructions.
Keyboard shortcut: Shift+F8.

Step Out

Executes the code out of a function call, and stops on the instruction immediately following the call to the function. Using this command, you can quickly finish executing the current function after determining that a bug is not present in the function.
Keyboard shortcut: Ctrl+Shift+F8.

Run to Cursor

When the VRML client is in break mode, use Run To Cursor to select a statement further down in your code where you want execution to stop. Your code will run from the current statement to the selected statement and the current line of execution margin indicator ➪ appears in the Selection margin.
Keyboard shortcut: Ctrl+F8.

Toggle Breakpoint

Sets or removes a breakpoint at the current line. Breakpoint is a selected line of code at which execution automatically stops. You can't set a breakpoint on lines containing nonexecutable code such as native VRML text, comments, or blank lines.
A line of code in which a breakpoint is set appears in the colors specified in the Format tab of the Options dialog box.
Keyboard shortcut: F9.

Clear All Breakpoints

Removes all breakpoints in your files. You cannot undo the Clear All Breakpoints command.

Show Next Statement

Highlights the next statement to be executed. Use the Show Next Statement command to place the cursor on the line that will execute next.

Quick Watch

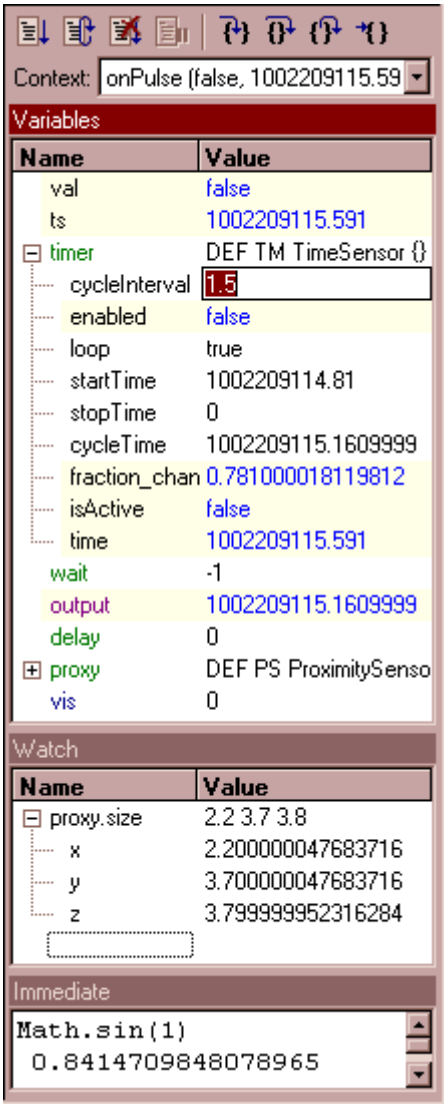
Displays the Quick Watch dialog box with the current value of the selected expression. Use this command to check or modify the current value of a variable, property, or other expression for which you have not defined a watch expression. Select the expression from Text editor and then choose the Quick Watch command. To add a watch expression based on the expression in the Quick Watch dialog box, choose the Add Watch button.
Keyboard shortcut: Shift+F9.

Modules

Displays the Modules dialog box with loaded to VRML client files, Script nodes and vrmlscript functions.

Debugger windows

Several specialized windows display debugging information for your code. When you are debugging, you can access these windows using the View menu. (In addition to windows, the debugger also uses dialog boxes to display debugging information.)



Context Box

The Context Box contains the current call stack in a drop-down list box. Use this list to specify the current scope of the variables displayed. This procedure changes the view of the code displayed in the Text editor, the Variables window and other debugger windows, but does not change the next line of execution or the value stored in the program counter.

Variables Window
The Variables window provides quick access to variables that are available in the current context and displays the variables that are local to the current function, interface fields and outgoing events of the Script node and global variables.
The Variables window contains a spreadsheet with fields for the variable name and value. The debugger automatically fills in these fields.
You cannot add variables or expressions to the Variables window (you must use the Watch window) - but you can expand or collapse the variables shown using the tree controls. You can expand an array or object variable in the Variables window if it has a plus sign (+) box in the Name field. If an array or object variable has a minus sign (–) box in the Name field, the variable is already fully expanded. To expand or collapse the variable, click the + or – box, as described in Spreadsheet Fields.
Watch Window
Use the Watch window to specify variables and expressions that you want to watch and modify while debugging your code.
If you add an array or object variable to the Watch window, plus sign (+) or minus sign (–) boxes appear in the Name column. You can use these boxes to expand or collapse your view of the variable, as described in Spreadsheet Fields.
Immediate Window
The Immediate window displays information from commands typed directly into the window. You can type or paste a line of code and press ENTER to run it. Use the Immediate window to:
- Test problematic or newly written code.
- Query or change the value of a variable. While execution is halted, assign the variable a new value as you would in code.
- Call procedures as you would in code.

Spreadsheet Fields

The debugger interface uses spreadsheet fields, which have an interface similar to that of Microsoft Excel. These spreadsheet fields appear in the Watch window, the Variables window, and the QuickWatch dialog box.

The window where the spreadsheet field is located determines which cells are editable. In the Variables window, you can edit the cells in the Value column. In the Watch window, you can edit the cells in the Name and Value columns.

Spreadsheet fields contain controls for easy viewing of array and object variables. If the variable is an array or object, the branch below the variable contains the component elements or properties.

The variables are marked with a box containing a plus sign (+) in the Name column. You can expand the variable by clicking the + box, which opens into a tree that may contain additional boxes. When a variable is expanded, the box in the Name column contains a minus sign (–). You can collapse an expanded variable by clicking the – box. As an alternative, you can expand a variable by selecting it and pressing the ENTER, PLUS SIGN or RIGHT ARROW key. You can collapse a variable by selecting it and pressing the ENTER, MINUS SIGN or LEFT ARROW key.

Scalar variables, which have no components to expand, do not have boxes in the Name column.

When working with these fields, you can size a column manually by dragging the divider at the right edge of the column.

Viewing the Value of a Variable

The easiest way to see the value of a variable when the debugger is stopped at a breakpoint is to use CodeTips pop-up information.

You can view a CodeTips pop-up information box for any variable that appears in a Text editor and is within the current scope. To see a pop-up box, place the mouse pointer over the variable.

CodeTips pop-up information is not available for invalid or undefined variables.

To view the value of a variable or expression:

1. Wait for the debugger to stop at a breakpoint, or
   Click **Break** or **Break in Script** on the **Debug** menu to halt the debugger.
2. On the **Debug** menu, click **QuickWatch**.
3. Type or paste the variable name or expression into the **Expression** text box.
4. Click **Close**.

Tip: The Expression drop-down list box contains the most recently used QuickWatch expressions.

To view the value of a variable using QuickWatch:

1. When the debugger is stopped at a breakpoint, switch to a source window, and click the right mouse button on a variable or selected expression (*Var*, for example).
2. On the shortcut menu, click **QuickWatch** *Var*.
3. Click the **Recalculate** button.
4. Click **Close**.

To view a variable or expression in the Watch window:

1. Start debugging, and pause the debugger in break mode (program is waiting for user input after completing a debugging command).
2. Type, paste, or drag the variable name or expression into the Name column on the **Watch** window.
3. Press ENTER (if typing).

The Watch window evaluates the variable or expression immediately and displays the value or an error message.

To view a variable in the Variables window:

1. Start debugging, and pause the debugger in break mode (program is waiting for user input after completing a debugging command).
2. Find the variable in the Name column and view its value.

To view a variable or expression in the Immediate window:

1. Start debugging, and pause the debugger in break mode (program is waiting for user input after completing a debugging command).
2. Type, paste, or drag the variable name or expression into the window.
3. Press ENTER.

The Immediate window evaluates the variable or expression immediately and displays the value or an error message.

Modifying the Value of a Variable

When the script is paused at a breakpoint or between steps, you can change the value of any vrmlscript variable in your code.

To modify the value of a variable using QuickWatch:

1. On the **Debug** menu, click **QuickWatch**.
2. In the **Expression** text box, type the variable name.
3. Click the **Recalculate** button.
4. If the variable is an array or object, use the + box to expand the view until you see the value you want to modify.
5. Double-click the value or use the TAB key to move to the value you want to modify.
6. Type the new value, and then press ENTER.
7. Click **Close**.

To modify the value of a variable using the Watch window:

1. If the variable is an array or object, use the + box to expand the view until you see the value you want to modify.
2. Double-click the value or use the TAB key to move the insertion point to the value you want to modify.
3. Type the new value, and press ENTER.

To modify the value of a variable in the Variables window:

1. Select the line containing the variable whose type you want to modify.
2. If the variable is an array or object, use the + box to expand the view until you see the value you want to modify.
3. Double-click the value, or use the TAB key to move the insertion point to the value you want to modify.
4. Type the new value, and press ENTER.

To modify the value of a variable in the Immidiate window:

1. Type the variable name or l- value expression, the assignment operator (**=**) and the new value.
2. Press ENTER.

Publishing Wizard

The Basics of Publishing

Publishing refers to the process of locating all the files necessary to create a document, scene, or world and organizing the files for publication on a server. The Publishing Wizard is also able to put the files on your Web server, using Microsoft Web Publishing engine, or send it by e-mail, using MAPI transports.

Publishing is a one-way process. Publishing leaves your original files alone, but as it makes copies of those files it changes and optimizes those copies. Any time you want to change your world, reopen the original source files for the world, then make changes and republish. Don't modify the published copies directly.

Setting Up the Destination

The first step of the publishing process is to choose a destination for your published files.

If you have already published your scene, choose an existing destination directory. Later in the publishing process, the Publishing Wizard may remove the contents of the directory before creating the new version of the published world.

If you haven't previously published your scene, enter a name for a new destination directory (which the Publishing Wizard then creates for you), or click the Browse button and choose a directory from the file browser.

If you have selected Publish files to the net option for the first time, get ready to download and install the latest version of Microsoft Web Publishing Wizard. It automates the process of copying files from a temporary destination directory to the Web or FTP server.

Click Next when you're done.

If you change your mind, you can come back to this dialog by clicking a Back button. At any point in the publishing process you can click Cancel to stop the process without publishing.

If you already published some scenes, you can click Finish to complete the process now using recently used settings for the rest of the dialogs.

Choosing Additional Resource Directories

The next step of the publishing process is to choose the additional resource directories. Some scenes may contain invalid links to the resources, that that are possibly kept somewhere. To help the Wizard find these resources, you can specify a list of directories. Starting from these directories, the Publishing Wizard will search missing resources.

To add a directory to the list:

1. Click the **Add** button.
2. Choose a directory from the file browser.
3. Click **OK**.

Click Next when you're done, or Back to return to the previous Wizard step.

Excluding Files from Publishing

This step lets you list files that you don't want the Publishing Wizard to include. You might want to exclude files that have been previously published, files that have been manually checked, files you know will already be on the Web server, or test files that you don't want to include in the final world.

Exclude filenames from the publish by selecting them in the list, then click Exclude. Revert this operation by selecting filenames in the list, then click Include. Excluded filename are marked by a cross mark .

URLs in your document which refer to these files are left unchanged in your documents.

Click Next when you're done, or Back to return to the previous Wizard step.

VRML Optimization Options

This step of the publishing process lets you choose options to optimize your world.

Pack VRML files using maximum compression

Compressed files download faster and use less disk space, so use of this feature is recommended. Most browsers automatically decompress files that have been compressed. To disable this feature, click the box to remove the check.

Remove extra formatting from VRML files

Check the box if you want the Publishing Wizard to reformat the VRML files by unindenting lines, removing unnecessary whitespaces and empty lines. Reformatted files use less disk space, but usually lost in human readability.

Remove comments

Selecting this option removes all comments from a file during the reformat.

Remove default field values

Selecting this option removes field values which are explicitly declared with a value identical to the default field value.

Simplify floating point numbers

Check the box if you want the Publishing Wizard when possible to reduce float numbers to integers without lost of quality, remove trailing zeros and remove unnecessary zero values to the left of decimal point. For example, number 10.0 reduce to 10, 0.10 – to .1 and so on.

Adjust numeric resolution

This option also removes unnecessary text, but in some cases such removal is lossy, and in other cases there may be external code, such as an EAI application, relying on the text. So you should be considered carefully before use the option.

This option adjusts numeric resolution of floating point numbers. In many cases, exporters output these numbers with a much greater resolution than necessary. The sub-options allow resolution to be indicated by type of the number. For example, coordinate resolution relates to all world coordinates and sizes: the point field in the Coordinate node, the size field in the Box node and so on. Orientation resolution relates to all SFRotation and MFRotation fields and also to the fields that defines angles.

Click Next when you're done, or Back to return to the previous Wizard step.

Specifying Directory Organization

In this step, you decide whether you want to use a flat directory structure, or whether you want to use a hierarchical structure.

Put all source files in one directory

This is the best choice if your world uses only a few files.

Arrange source files in subfolders, depending of its type

Use the resource type based directory structure. This is the best choice if you have a complex multi-file world.

Retain the same directory structure used by the source files

Use the same directory structure that your source files use. This is the best choice if you have a complex multi-file world stored in a set of nested directories.

In-line resources as base64 encoded data

Include resources of specified types into VRML files body, excluding it from the list of publishing files. VRML files with in-lined resources are currently supported only by Cortona VRML viewer.

Click Next when you're done, or Back to return to the previous Wizard step.

Reviewing the Directory Structure

In this step, you review the directory structure. This dialog lists every file that will be placed in the destination directory. If the files don’t have the names and locations you expect, click Back to go back to the mappings dialog.

Click Next when you're done, or Back to return to the previous Wizard step. Your files are set up for final publishing when you click Next, so you can't turn back to previous steps from this point.

Previewing Published Documents

The final step of the Publishing lets you preview the published world immediately. Click Preview to preview the world in your default browser. Click Open Folder to open the destination directory with your files, using Windows Explorer window.

If you have chosen publishing to the net or through e-mail in the first step, click Next to launch Microsoft Web Publishing Wizard or mail sender. Otherwise, click Close to finish publishing.

Hints and Tips

How to compress a VRML file

Compressed files download faster and use less disk space, so use of this feature is recommended. Most browsers automatically decompress files that have been compressed.

To compress an existing VRML file:

1. From the **File** menu, choose **Open**.
2. Select your file.
3. Click **OK**.
4. Change the compression level, if you like, as written below.
5. From the **File** menu, choose **Save As**.
6. Select **Save compressed** checkbox.
7. Enter or select a destination file name.
8. Click **OK**.

To change default compression level:

1. From the Tools menu, choose Options.
2. Select the **General** tab.
3. Use the **Default compression** drop-down list to select the compression level:
   - Maximum – best compression, but lowest speed of compression;
   - Normal – balance between speed and compression quality;
   - Fast – fastest speed, but poor compression quality.
4. Click **OK**.

To compress a file without using the Save As dialog box:

1. Double-click the pane **GZ** in the status bar at the bottom of VrmlPad window.
2. From the **File** menu, choose **Save**, or type Ctrl+S.

How to download a VRML file from the Net.

If you are using download dialog for the first time:

1. From the Tools menu, choose Options.
2. Select the **General** tab.
3. Enter or browse destination directory in the **Location for downloaded files** box.

   This directory will keep the downloaded files and its resources – images, audio files, scripts, etc.
4. Specify additional download options.
5. Click **OK**.

To download a file:

1. From the **File** menu, choose **Download**, or double-click the pane **WEB** in the status bar at the bottom of VrmlPad window.
2. Type valid Internet address (URL) of your file to the **Open** box, or use the drop-down list to select previous URLs. When you select an URL in this list, the URL is added in the **Open** box.
3. Specify additional download options.
4. Click **OK**.

To download a file without using the Download dialog box:

- Drag-and-drop an Internet shortcut from your Web-browser to the caption of VrmlPad.

To cancel downloading:

- Double-click the animated globe in the status bar at the bottom of VrmlPad window.

How to upload a VRML file to a remote server.

First of all you should download and install the latest version of Microsoft Web Publishing Wizard from http://www.microsoft.com/windows/software/webpost/. It automates the process of copying files from your PC to the Web or FTP server.

To upload a VRML file using Microsoft Web Publishing Wizard:

1. Open the file into VrmlPad.
2. From the **File** menu, choose **Publish**.
3. Select **Publish files to the net** options in the first page.
4. Proceed through the rest of the Publishing Wizard.
5. After the last page, follow instructions of the Microsoft Web Publishing Wizard. For the first time, specify path to your FTP server, user name and login.

How to quickly find the definition (reference) of an identifier.

To jump to the definition of an identifier (PROTO, node, or field name):

1. Position the caret at the identifier in the Text editor, or select an identifier in the Scene Tree.
2. Select **Go To Definition** from the pull-down menu or type Ctrl+F11.

To jump to the reference of an identifier (PROTO, node, or field name):

1. Position the caret at the identifier in the Text editor, or select an identifier in the Scene Tree.
2. Select **Go To Reference** from the pull-down menu or type F11.
3. Select **Next Reference** from the pull-down menu or type F11 to jump to the next reference of the identifier.

How to insert a pair of node or PROTO braces.

To insert a pair of braces after node or PROTO name:

1. Position the caret to the space after node or PROTO name.
2. Type Ctrl+Space.

How to insert default field value.

To insert a default value after field declaration or definition:

1. Position the caret to the space after the field declaration or definition or inside multiple field value.
2. Type Ctrl+Space.

How to indent a block of lines.

To indent a block of lines:

- Select the lines and then press Tab.

To unindent a block of lines:

- Select the lines and press Shift+Tab.

To format a block of lines using the smart indent settings:

- Select the lines and press Ctrl+Shift+F.

How to comment a block of lines or an entire node.

To comment a block of lines:

- Select the lines and press Ctrl+Shift+C.

To uncomment a block of lines:

- Select the commented lines and press Ctrl+Shift+C.

To comment a node, PROTO declaration or field:

- Select the identifier of a node, PROTO declaration or field in the Scene Tree and choose the **Comment** from the pull-down menu or press Ctrl+Shift+C.

How to locate a syntax or semantic error.

To jump to the first error:

- Double-click on the highlighted error indicator in the status bar at the bottom of the VrmlPad window.

To jump to the next error:

- From the **View** menu, choose **Next Error** or type F4.

How to make a VRML file smaller.

Reducing file size and improving reliability and performance go hand in hand. VrmlPad can automatically remove code that is unnecessary, redundant or unusable. The size reduction is often dramatic, and the resulting file is more likely to run consistently across browsers.

To reduce file size using the Unused Identifiers dialog box:

1. From the **View** menu, choose **Unused Identifiers**.
2. In the list below, select identifiers you want to remove.
3. Click **Delete** or type Del.

   Note: You can savely remove default field values. Remove identifiers of the other types with caution – some node names or field declarations may be used by outer scripts, EAI or ActiveX components. Also, some VRML authoring tools may use unnecessary for rendering node names and field declarations for holding additional user interface information, such as object names and properties.

To reduce file size using the Publishing Wizard:

1. From the **File** menu, choose **Publish**.
2. In the first page, choose copying document to a folder.
3. Proceed through the following pages up to the **Specify VRML preferences** page.
4. Select the options you want to.
5. Proceed through the rest of the Wizard.

How to expose a field to a PROTO interface.

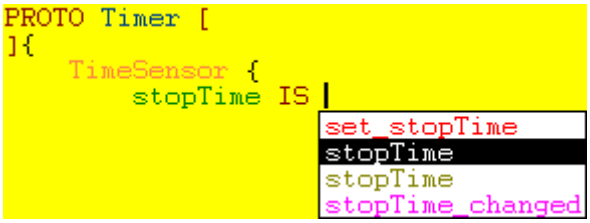
To automatically expose a field:

1. Position the caret to the space after the field definition or **Script** node's field declaration (without a field value).
2. Type the **IS** keyword into your code and press SPACE.
3. Type Ctrl+Space and select one of the choices.

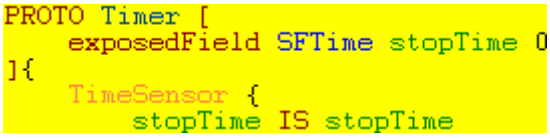
   There are four possible choices in the case of the exposedField:

   - Expose the field as an exposeField declaration.

- Expose the field as a field declaration.
- Expose the field as an eventIn declaration.
- Expose the field as an eventOut declaration.



4. Press SPACE or ENTER.



How to register a VRML extension.

VRML extension is a custom node type in addition to a set of the standard VRML97 nodes.

To register a VRML extension:

1. Create a VRML file (in VrmlPad, for instance), containing PROTO declarations with empty bodies. These PROTOs should describe your VRML extension – names of the additional node types, field names, categories and types.

   Note: Actually, a set of the standard VRML97 nodes implemented internally in VrmlPad as a hidden VRML file. To open it in the editor, type Ctrl+Shift+N.
2. Save the file somewhere.
3. From the **Tools** menu, choose **Options**.
4. Select the **General** tab.
5. Select **Enable VRML extensions** checkbox.
6. Type in the **Source** field path to the saved file or browse it using a button at the right of the field.
7. Click **OK**.
8. Restart VrmlPad.

How to change a set of the standard nodes.

To change a standard set of node types, using VrmlPad:

1. Press Ctrl+Shift+N to load the standard nodes as PROTO declarations.
2. Make all changes you want to.
3. Save the file somewhere.

4. From the **Tools** menu, choose **Options**.
5. Select the **General** tab.
6. Select **Enable VRML extensions** checkbox.
7. Type in the **Source** field path to the saved file or browse it using a button at the right of the field.
8. Select **Ignore all standard node types** checkbox.
9. Click **OK**.
10. Restart VrmlPad.

How to change a source of the Node Help.

Help for the standard VRML97 nodes is available through the Help for Current Node from the Help menu. By default, VrmlPad uses remote HTML manual http://www.web3d.org/technicalinfo/specifications/vrml97/part1/nodesRef.html, but you can download the manual into your PC and redirect VrmlPad on it.

To change a Node Help source:

1. From the Windows **Start** menu, choose **Run**.
2. In the **Open** box, type ‘regedit’, and click **OK**.
3. Open registry key HKEY_CURRENT_USER/Software/ParallelGraphics/VrmlPad/Settings.
4. Create a new string value, named ‘NodesRef’.
5. Specify in this value full path to your downloaded manual, for example – ‘C:\VRML Spec\part1\nodesRef.html’.

What are the ways I can access the VrmlPad object model?

You can access the VrmlPad object model in three ways:

- Write VBScript or JavaScript macros that script VrmlPad through its object model.
- Create add-ins.
- In ActiveX-enabled applications, write an Automation controller that accesses VrmlPad by creating an instance of it. For example, in Visual Basic, use the CreateObject function to create an instance of VrmlPad. Or in C++, call the Win32 CoCreateInstance function. For more information, see the application's documentation.

How to automate VrmlPad from another application.

With Automation, you can control VrmlPad from other applications such as Word, Excel, or Visual Basic. You can start VrmlPad, automate VrmlPad tasks, and then stop VrmlPad when the tasks are finished.

To start VrmlPad from another application, create an instance of the VrmlPad Document object. For example, to start VrmlPad from Visual Basic or Visual Basic for Applications, use the following code:

```
Dim doc As Object
Set doc = CreateObject("VrmlPad.Document")
```

In Visual Basic you can use the VrmlPad type library to access the VrmlPad object model. To do this, from the Project menu choose References and select the *VrmlPad Application Library* checkbox.

```
Dim doc As VrmlPad.Document
Set doc = New VrmlPad.Document
```

When VrmlPad starts, it will be invisible — unless you make it visible by using the Visible property of the Window object:

```
Dim doc As VrmlPad.Document
Set doc = New VrmlPad.Document
doc.Window.Visible = True
```

After starting VrmlPad, you can automate tasks in it by using the appropriate code. For example, in the following code, a *WorldInfo* node is added to the created document:

```
Dim doc As VrmlPad.Document
Set doc = New VrmlPad.Document
doc.Window.Visible = True
doc.RootNodes.Add("WorldInfo")("info") = "Created by me"
```

After completing your VrmlPad tasks, you can stop VrmlPad releasing all references to the Document object. If you use Visual Basic to automate VrmlPad, Visual Basic automatically releases all references to the Document object.

Note: If the user tries to stop VrmlPad while your program is controlling it, VrmlPad will remain in memory but the user interface will become invisible. If this happens, release all references to the Document object so that VrmlPad can quit.